Relational Parametricity for Control Considered as a Computational Effect

Mogelberg, R., & Simpson, A. (2007). Relational Parametricity for Control Considered as a Computational Effect. Electronic Notes in Theoretical Computer Science, 173, 295-312doi: 10.1016/j.entcs.2007.02.040

Relational Parametricity for Control Considered as a Computational Effect

Rasmus Ejlers Møgelberg, Alex Simpson[1]

LFCS, University of Edinburgh

Abstract

This paper investigates parametric polymorphism in the presence of control operators. Our approach is to specialise a general type theory combining polymorphism and computational effects, by extending it with additional constants expressing control. By defining relationally parametric models of this extended calculus, we capture the interaction between parametricity and control. As a worked example, we show that recent results of M. Hasegawa on type definability in the second-order (call-by-name) $\lambda\mu$-calculus arise as special cases of general results valid for arbitrary computational effects.

Keywords: Computational effects, control, denotational semantics, parametric polymorphism

1 Introduction

Relational parametricity [19] gives a powerful proof principle for establishing properties of polymorphic programs. It has been widely studied in the context of the Girard/Reynolds second-order λ-calculus, which, in its pure form, is a calculus of total functions.

Trying to extend the notion of relational parametricity to richer calculi causes problems. Even recursion (and attendant nontermination) create difficulties, since the fixed-point property of recursion is incompatible with certain consequences of parametricity (the existence of finite coproducts). This issue led Plotkin to advocate adopting a linear type theory, in which such inconsistencies do not arise [18]. This approach has been worked out in detail in [4], and enjoys many good properties. For example, one obtains a wide class of polymorphic type definability results, with the desired universal properties following from relational parametricity.

Unfortunately, the approach of using linear type theory does not adapt to impure settings incorporating computational effects. In particular, a crucial aspect of linear logic, its monoidal closed structure, does not occur naturally in models

[1] This work was supported by the Danish Agency for Science, Technology and Innovation, and by EPSRC

for computational effects whose associated computational monads (in the sense of Moggi [15]) are not commutative.

One important example of a non-commutative monad is the continuations monad used to model control effects. For such effects, Hasegawa [6,7] has recently proposed a syntactic account of relational parametricity, and he has exploited this to obtain type definability results for a call-by-name version of Parigot's second-order $\lambda\mu$-calculus [16]. An intriguing fact he observes is that his results on polymorphic type definability are analogous to ones that hold in Plotkin's linear setting. However, the technical frameworks within which the two classes of results are developed are seemingly quite different.

The goal of the present paper, together with a companion paper [14], is to develop a unified semantic theory of relational parametricity, which is applicable to arbitrary computational effects, and which specialises, in the case of commutative effects, to linear parametricity, and, in the case of control, to a semantic version of Hasegawa's account of parametricity.

The general framework we use to achieve this is presented in detail in the companion paper [14]. It takes the form of a polymorphic type theory loosely based on Levy's call by push value paradigm [13], together with an associated semantic framework for defining relationally parametric models.

In the current paper, we focus on control. To this end, we extend the general type theory of [14] with extra constants which specialise it to control. The extension allows control-based program calculi to be interpreted in the type theory. We illustrate this by giving a call-by-name translation of Parigot's second-order $\lambda\mu$-calculus, closely following the formulation in [7].

The main technical effort in the paper is the construction of a relationally parametric model of the extended type theory. Our goal is to exploit the model to show that Hasegawa's results on type definability in the second-order $\lambda\mu$-calculus also hold in our setting. In fact, we show that they arise as special cases of general results valid for arbitrary computational effects. As well as providing an alternative (and denotational) account of Hasegawa's results, this fact explains the reason that Hasegawa's results mirror ones available in the setting of linear parametricity.

2 A type theory for polymorphism and control

We start by recalling the general type theory for polymorphism and effects (PE) as presented in [14], and specialising it to the case of continuations. The type theory for polymorphism and effects is based on Paul Levy's call-by-push-value (CBPV) paradigm [13], in which types are separated into two groups: *value types* and *computation types*. Following the convention of [13], we distinguish syntactically between the two kinds of type by using underlined metavariables $\underline{\mathsf{A}}, \underline{\mathsf{B}}, \ldots$ for computation types, as opposed to $\mathsf{A}, \mathsf{B}, \ldots$ for value types. The type theory PE allows for polymorphic quantification over both kinds of type, and therefore types may contain type variables for value type, ranged over by $X, Y, \ldots$ or for computation types, ranged over by $\underline{X}, \underline{Y}, \underline{Z}, \ldots$. Value types and computation types are defined by the

grammar

$$\mathsf{A}, \mathsf{B} ::= X \mid \mathsf{A} \to \mathsf{B} \mid \forall X.\, \mathsf{A} \mid \underline{X} \mid \underline{\mathsf{A}} \multimap \underline{\mathsf{B}} \mid \forall \underline{X}.\, \mathsf{A}$$

$$\underline{\mathsf{A}}, \underline{\mathsf{B}} ::= \mathsf{A} \to \underline{\mathsf{B}} \mid \forall X.\, \underline{\mathsf{A}} \mid \underline{X} \mid \forall \underline{X}.\, \underline{\mathsf{A}} \ .$$

Notice that, in contrast to CBPV, our computation types constitute a subset of the value types. For discussion on this and other departures from CBPV, see [14]. Here we comment only on those features that will play a prominent role in our treatment of control.

There are two constructions for function types: $\multimap$ and $\to$, and to distinguish the two we shall refer to the former as the *linear* function type. One semantic intuition of the type system is that computation types are algebras for a monad and value types are simply objects. Under this intuition $\to$ denotes ordinary function space and $\multimap$ is the space of algebra homomorphisms, which does not in general carry an algebra structure so is a value type. Another somewhat different model will be presented in Section 4. Note that typing restrictions prevent the $\multimap$ constructor from being iterated. For example, $(\underline{\mathsf{A}} \multimap \underline{\mathsf{B}}) \multimap \underline{\mathsf{B}'}$ is not a valid type.

Terms are given by the grammar

$$t, s ::= x \mid \lambda x\colon \mathsf{A}.\, t \mid s(t) \mid \Lambda X.\, t \mid t(\mathsf{A}) \mid \lambda^{\circ} x\colon \underline{\mathsf{A}}.\, t \mid \Lambda \underline{X}.\, t$$

Typing judgements are of the form $\Gamma \mid \Delta \vdash t\colon \mathsf{A}$, where Γ is a context of variables and Δ is a second variable context called the *stoup* which is either empty or consists of exactly one variable of computation type, in which case A must also be a computation type. The two cases are also sometimes written explicitly as

$$\Gamma \mid - \vdash t\colon \mathsf{A} \qquad\qquad \Gamma \mid x\colon \underline{\mathsf{B}} \vdash t\colon \underline{\mathsf{A}} \ .$$

The semantic intuition behind the stoup is that the term t, for all instantiations of the variables in Γ, denotes an algebra homomorphism from $\underline{\mathsf{B}}$ to $\underline{\mathsf{A}}$. This is closely related to Levy's notion of *stack* in CBPV [13].

The typing rules are given in Figure 1. We shall consider the equality theory on terms of the type theory given as the smallest congruence relation induced by the usual β and η rules for both types of term abstraction and both types of type abstraction, see Figure 2. Of course these equalities are to be understood as being between identically typed terms in the relevant contexts.

In [14], the type theory PE is studied as a generic calculus for combining polymorphism and effects. Our purpose here is to specialise it to the case of control effects, as implemented by continuations monads. To motivate our approach, we recall that, in [14], the defined type construct

$$!\mathsf{A} \ =_{\mathrm{def}} \ \forall \underline{X}.\, (\mathsf{A} \to \underline{X}) \to \underline{X} \ ,$$

which maps any value type A to a computation type $!\mathsf{A}$, is shown to serve the purpose of Moggi's type monadic type $T\mathsf{A}$ [15] (or, slightly more accurately, of Levy's free computation type construct $F\mathsf{A}$ [13]). In [14], a theory of relational parametricity is used to justify this encoding. As a consequence, one obtains a Girard decomposition: the types $\mathsf{A} \to \underline{\mathsf{B}}$ and $!\mathsf{A} \multimap \underline{\mathsf{B}}$ are isomorphic (as value types).

$$\frac{}{\Gamma, x\colon \mathsf{A} \mid - \vdash x\colon \mathsf{A}} \qquad \frac{\Gamma, x\colon \mathsf{A} \mid \Delta \vdash t\colon \mathsf{B}}{\Gamma \mid \Delta \vdash \lambda x\colon \mathsf{A}.\, t\colon \mathsf{A} \to \mathsf{B}} \qquad \frac{\Gamma \mid \Delta \vdash s\colon \mathsf{A} \to \mathsf{B} \quad \Gamma \mid - \vdash t\colon \mathsf{A}}{\Gamma \mid \Delta \vdash s(t)\colon \mathsf{B}}$$

$$\frac{\Gamma \mid \Delta \vdash t\colon \mathsf{A}}{\Gamma \mid \Delta \vdash \Lambda X.\, t\colon \forall X.\, \mathsf{A}}\ X \notin \mathrm{FTV}(\Gamma, \Delta) \qquad \frac{\Gamma \mid \Delta \vdash t\colon \forall X.\, \mathsf{A}}{\Gamma \mid \Delta \vdash t(\mathsf{B})\colon \mathsf{A}[\mathsf{B}/X]}$$

$$\frac{}{\Gamma \mid x\colon \underline{\mathsf{A}} \vdash x\colon \underline{\mathsf{A}}} \qquad \frac{\Gamma \mid x\colon \underline{\mathsf{A}} \vdash t\colon \underline{\mathsf{B}}}{\Gamma \mid - \vdash \lambda^{\circ} x\colon \underline{\mathsf{A}}.\, t\colon \underline{\mathsf{A}} \multimap \underline{\mathsf{B}}} \qquad \frac{\Gamma \mid - \vdash s\colon \underline{\mathsf{A}} \multimap \underline{\mathsf{B}} \quad \Gamma \mid \Delta \vdash t\colon \underline{\mathsf{A}}}{\Gamma \mid \Delta \vdash s(t)\colon \underline{\mathsf{B}}}$$

$$\frac{\Gamma \mid \Delta \vdash t\colon \mathsf{A}}{\Gamma \mid \Delta \vdash \Lambda \underline{X}.\, t\colon \forall \underline{X}.\, \underline{\mathsf{A}}}\ \underline{X} \notin \mathrm{FTV}(\Gamma, \Delta) \qquad \frac{\Gamma \mid \Delta \vdash t\colon \forall \underline{X}.\, \mathsf{A}}{\Gamma \mid \Delta \vdash t(\underline{\mathsf{B}})\colon \mathsf{A}[\underline{\mathsf{B}}/X]}$$

Fig. 1. Typing rules for PE.

$$\begin{aligned}
(\lambda x\colon \mathsf{A}.\, t)(u) &= t[u/X] & \\
\lambda x\colon \mathsf{A}.\, t(x) &= t & \text{if } t\colon \mathsf{A} \to \mathsf{B} \text{ and } x \notin \mathrm{FV}(t) \\
(\lambda^{\circ} x\colon \mathsf{A}.\, t)(u) &= t[u/X] & \\
\lambda^{\circ} x\colon \mathsf{A}.\, t(x) &= t & \text{if } t\colon \mathsf{A} \multimap \mathsf{B} \text{ and } x \notin \mathrm{FV}(t) \\
(\Lambda \underline{X}.\, t)\, \underline{\mathsf{A}} &= t[\underline{\mathsf{A}}/\underline{X}] & \\
\Lambda \underline{Y}.\, t\, \underline{Y} &= t & \text{if } t\colon \forall \underline{X}.\, \mathsf{A} \text{ and } \underline{Y} \notin \mathrm{FTV}(t) \\
(\Lambda X.\, t)\, \mathsf{A} &= t[\mathsf{A}/X] & \\
\Lambda Y.\, t\, Y &= t & \text{if } t\colon \forall X.\, \mathsf{A} \text{ and } Y \notin \mathrm{FTV}(t)
\end{aligned}$$

Fig. 2. Equality axioms for PE.

To encode control, we wish to specialise $!\mathsf{A}$ to be a continuations monad. One can formulate this by asking for a type constant R, to act as a result type, and requiring the type construction $(\mathsf{A} \to R) \to R$ to behave like $!\mathsf{A}$. For this, the type expression $(\mathsf{A} \to R) \to R$ must be a computation type, hence so must the type constant R. Then the required properties of $(\mathsf{A} \to R) \to R$ can be implemented by requiring the canonical linear map $!\mathsf{A} \multimap (\mathsf{A} \to R) \to R$ to have a linear inverse. By the Girard decomposition above, this is equivalent to asking for the canonical map $!\mathsf{A} \multimap (!\mathsf{A} \multimap R) \to R$ to have a linear inverse.

Our actual approach to control is a natural generalisation of the above. Rather than restricting the last isomorphism above to types of the form $!\mathsf{A}$, we ask for the canonical map $\underline{\mathsf{A}} \multimap (\underline{\mathsf{A}} \multimap R) \to R$ to be an isomorphism for every computation type $\underline{\mathsf{A}}$. Of course, the natural way to add such an isomorphism to the type theory is by adding a polymorphic constant. One direction of the isomorphism is directly definable without any additions to the type theory:

$$\eta =_{\mathrm{def}} \Lambda \underline{X}.\, \lambda^{\circ} x\colon \underline{X}.\, \lambda f\colon \underline{X} \multimap R.\, f(x)\colon \forall \underline{X}.\, \underline{X} \multimap (\underline{X} \multimap R) \to R\ .$$

We thus extend the type theory with a constant

$$\epsilon\colon \forall \underline{X}.\, ((\underline{X} \multimap R) \to R) \multimap \underline{X}\ ,$$

and equations ensuring that this constant is inverse to η. More precisely, using the notation $\epsilon_{\underline{\mathsf{A}}}$ for $\epsilon\,\underline{\mathsf{A}}$ and likewise $\eta_{\underline{\mathsf{A}}}$ for $\eta\,\underline{\mathsf{A}}$, we add the equations

$$\Lambda\underline{X}.\,\lambda^{\circ}x\colon \underline{X}.\,\epsilon_{\underline{X}}(\eta_{\underline{X}}(x)) = \Lambda\underline{X}.\,\lambda^{\circ}x\colon \underline{X}.\,x$$
$$\Lambda\underline{X}.\,\lambda^{\circ}x\colon (\underline{X}\multimap R)\to R.\,\eta_{\underline{X}}(\epsilon_{\underline{X}}(x)) = \Lambda\underline{X}.\,\lambda^{\circ}x\colon (\underline{X}\multimap R)\to R.\,x.$$

We shall refer to this extension as the type theory for polymorphism and control, abbreviated as PE+C. Further motivation for the formulation of this theory can be found in Section 3, where it is used to model Parigot's second-order $\lambda\mu$-calculus [16], and in Section 4, where the type constants and equations will be justified semantically. We also mention that our use of the linear function space $\underline{\mathsf{A}}\multimap R$ is related to Levy's use of stack types in his treatment of control in [13, §5.4], and our use of R is related to his use of an answer type in [13, Ch. 7]. Our combination of the two seems a very natural approach, since the requirement of having a linear isomorphism between $\underline{\mathsf{A}}$ and $(\underline{\mathsf{A}}\multimap R)\to R$ leads to a very simple equational theory.

We now explore some simple equational properties of PE+C that follow from the $\beta\eta$ laws and the isomorphism equations for control. First, the constant ϵ is natural in linear maps in the sense that for any $f\colon \underline{\mathsf{A}}\multimap\underline{\mathsf{B}}$, the diagram

$$\begin{array}{ccc}
(\underline{\mathsf{A}}\multimap R)\to R & \xrightarrow{\epsilon_{\underline{\mathsf{A}}}}\circ & \underline{\mathsf{A}} \\
\Big|(f\multimap R)\to R & & \Big|f \\
\circ & & \circ \\
(\underline{\mathsf{B}}\multimap R)\to R & \xrightarrow{\epsilon_{\underline{\mathsf{B}}}}\circ & \underline{\mathsf{B}}
\end{array}$$

commutes. This follows from the corresponding naturality of η, which is a simple consequence of the β and η rules.

Next, we show that, just from the $\beta\eta$ equalities, one obtains a Girard decomposition of function spaces using $(\mathsf{A}\to R)\to R$ in place of $!\mathsf{A}$. (Note that the corresponding result for the polymorphic definition of $!\mathsf{A}$ depends upon parametricity properties.) For readability we introduce the notation $\neg\mathsf{A}$ for $\mathsf{A}\to R$.

Proposition 2.1 *For any type* A *and computation type* $\underline{\mathsf{B}}$ *in* PE+C *there is a bijective correspondence between terms of type* $\mathsf{A}\to\underline{\mathsf{B}}$ *and terms of type* $\neg\neg\mathsf{A}\multimap\underline{\mathsf{B}}$. *The correspondence is natural in* A *for all maps and in* $\underline{\mathsf{B}}$ *for linear maps.*

Proof. The correspondence takes a term $t\colon \neg\neg\mathsf{A}\multimap\underline{\mathsf{B}}$ to $\lambda x\colon \mathsf{A}.\,t(\lambda f\colon \neg\mathsf{A}.\,f(x))$ and a term $u\colon \mathsf{A}\to\underline{\mathsf{B}}$ to $\lambda^{\circ}h\colon \neg\neg\mathsf{A}.\,\epsilon_{\underline{\mathsf{B}}}(\lambda f\colon \underline{\mathsf{B}}\multimap R.\,h(f\circ u))$. □

An important consequence of adding the constant ϵ to the type theory is that η becomes split mono, which implies the following lemma.

Lemma 2.2 *In* PE+C *the following principle holds. If* $t,t'\colon \underline{\mathsf{A}}$ *are such that* $f(t) = f(t')$, *for a fresh variable* $f\colon \underline{\mathsf{A}}\multimap R$, *then* $t = t'$.

We end this section with another lemma that we need later.

Lemma 2.3 *In* PE+C *if* $f\colon \underline{A} \multimap R$ *and* $x\colon (\underline{A} \multimap R) \to R$ *then*

$$x(f) = f(\epsilon_{\underline{A}}\, x)$$

Proof. Naturality of ϵ implies $f(\epsilon_{\underline{A}}\, x) = \epsilon_R(((f \multimap R) \to R)(x))$. Now, by uniqueness of inverses to isomorphisms, $\epsilon_R(y) = y(id_R)$, and so $f(\epsilon_{\underline{A}} x) = x(f)$ as desired.□

3 Interpreting second-order $\lambda\mu$-calculus in PE+C

The goal of this section is to show how the control primitives of PE+C support a natural interpretation of Parigot's second-order $\lambda\mu$-calculus [16]. For this, we give a call-by-name translation, modelling the equational theory of [7].

We first recall the second-order $\lambda\mu$-calculus, $\lambda\mu 2$. Generally, we follow the presentation in [7], but extend the language slightly by introducing a falsity type, which allows us to break the control operations into name application and μ-abstraction. This is not a serious modification of the calculus. As shown in [7], see also Section 5, under an appropriate theory of relational parametricity, $\bot$ can be encoded polymorphically as $\forall X.\, X$.

The types of $\lambda\mu 2$ are given by the grammar

$$\sigma, \tau ::= X \mid \bot \mid \sigma \to \tau \mid \forall X.\, \sigma,$$

and terms are given by

$$M, N ::= x \mid M\ N \mid \lambda x^{\sigma}.\, M \mid M\ \sigma \mid \Lambda X.\, M \mid \mu\alpha.\, M \mid [\alpha]M.$$

We use x, y, z to range over ordinary variables and α, β, γ to range over continuation variables (sometimes called names). Typing judgements are written as

$$\Gamma \vdash M\colon \sigma \mid \Delta,$$

where Γ is the context of ordinary variables, and Δ is a context of continuation variables. The typing rules of $\lambda\mu 2$ are given in Figure 3.

$$\Gamma, x\colon \sigma, \Gamma' \vdash x\colon \sigma \mid \Delta \qquad \frac{\Gamma, x\colon \sigma, \Gamma' \vdash M\colon \tau \mid \Delta}{\Gamma, \Gamma' \vdash \lambda x^{\sigma}.\, M\colon \sigma \to \tau \mid \Delta}$$

$$\frac{\Gamma \vdash M\colon \sigma \to \tau \mid \Delta \qquad \Gamma \vdash N\colon \sigma \mid \Delta}{\Gamma \vdash M\ N\colon \tau \mid \Delta} \qquad \frac{\Gamma \vdash M\colon \sigma \mid \Delta \qquad X \notin \mathrm{FTV}(\Gamma, \Delta)}{\Gamma \vdash \Lambda X.\, M\colon \forall X.\, \sigma \mid \Delta}$$

$$\frac{\Gamma \vdash M\colon \forall X.\, \sigma \mid \Delta}{\Gamma \vdash M\ \tau\colon \sigma[\tau/X] \mid \Delta}$$

$$\frac{\Gamma \vdash M\colon \sigma \mid \Delta, \alpha\colon \sigma, \Delta'}{\Gamma \vdash [\alpha]M\colon \bot \mid \Delta, \Delta'} \qquad \frac{\Gamma \vdash M\colon \bot \mid \Delta, \alpha\colon \sigma, \Delta'}{\Gamma \vdash \mu\alpha^{\sigma}.\, M\colon \sigma \mid \Delta, \Delta'}$$

Fig. 3. Typing rules for $\lambda\mu 2$

We shall consider the equality relation on $\lambda\mu 2$ terms given as the smallest congruence relation containing the axioms of Figure 4. The two last rules in Figure 4 use so-called mixed substitution. For example $M[[\beta](-N)/[\alpha](-)]$ means replace all subterms of M of the form $[\alpha]L$ with $[\beta](L\,N)$.

$$\begin{aligned}
(\lambda x^{\sigma}.\, M)\, N &= M[N/x] \\
\lambda x^{\sigma}.\, M\, x &= M \\
(\Lambda X.\, M)\, \sigma &= M[\sigma/X] \\
\Lambda X.\, M\, X &= M \\
[\alpha](\mu\beta.\, M) &= M[\alpha/\beta] \\
\mu\alpha.\, [\alpha]M &= M (\alpha \notin \mathrm{FN}(M)) \\
(\mu\alpha^{\sigma\to\tau}.\, M)\, N &= \mu\beta^{\tau}.\, M[[\beta](-N)/[\alpha](-)] \\
(\mu\alpha^{\forall X.\sigma}.\, M)\, \tau &= \mu\beta^{\sigma[\tau/\alpha]} M[[\beta](-\,\tau)/[\alpha](-)]
\end{aligned}$$

Fig. 4. Axioms for call-by-name $\lambda\mu 2$

Our interpretation, denoted $(-)^*$, of $\lambda\mu 2$ into PE+C is presented in Figure 5. Types of $\lambda\mu 2$ are interpreted as computation types. A $\lambda\mu 2$ typing context $\Gamma = x_1 \colon \sigma_1, \ldots x_n \colon \sigma_n$ is interpeted as $\Gamma^* =_{\mathrm{def}} x_1 \colon {\sigma_1}^*, \ldots x_n \colon {\sigma_n}^*$. For a continuation context, $\Delta = \alpha_1 \colon \sigma_1', \ldots \alpha_m \colon \sigma_m'$, we use the notation $\Delta^* \multimap R$ for the PE+C context $\alpha_1 \colon {\sigma_1}^* \multimap R, \ldots \alpha_n \colon {\sigma_1}^* \multimap R$. The proposition below formulates the interpretation of typing judgements.

$$X^* = \underline{X} \qquad \bot^* = R \qquad (\sigma \to \tau)^* = \sigma^* \to \tau^* \qquad (\forall X.\, \sigma)^* = \forall \underline{X}.\, \sigma^*$$

$$x^* = x \qquad (\lambda x \colon \sigma.\, M)^* = \lambda x \colon \sigma^*.\, M^* \qquad (M\, N)^* = M^*\, N^*$$
$$(\Lambda X.\, M)^* = \Lambda \underline{X}.\, M^* \qquad (M\, \sigma)^* = M^* \sigma^*$$
$$([\beta]M)^* = \beta(M^*) \qquad (\mu\alpha^{\sigma}.\, M)^* = \epsilon_{\sigma^*}(\lambda\alpha \colon \sigma^* \multimap R.\, M^*)$$

Fig. 5. The interpretation of $\lambda\mu 2$ into PE+C.

Proposition 3.1 (Type soundness) *If $\Gamma \vdash M \colon \sigma \mid \Delta$ is a typed term in $\lambda\mu 2$ calculus then the judgement below is well typed in* PE+C.

$$\Gamma^*, \Delta^* \multimap R \mid - \vdash M^* \colon \sigma^*$$

Theorem 3.2 (Soundness) *If $\Gamma \vdash M_1 = M_2 \colon \sigma \mid \Delta$ is provable using the equality rules of call-by-name $\lambda\mu 2$ then* PE+C *proves:*

$$\Gamma^*, \Delta^* \multimap R \mid - \vdash M_1^* = M_2^* \ .$$

Proof. The proof is by induction over the structure of the proof of $M_1 = M_2$. The interpretation is clearly sound with respect to the first half of the $\lambda\mu 2$ axioms as the interpretation preserves the simply typed structure and the polymorphic structure.

Consider the equality $[\beta](\mu\gamma.\, M) = M[\beta/\gamma]$. By definition $([\beta](\mu\gamma.\, M))^* = \beta(\epsilon_{\sigma^*}(\lambda\gamma.\, M^*))$, where σ is the type of γ, which by Lemma 2.3 is equal to $M^*[\beta/\gamma] = (M[\beta/\gamma])^*$.

The rule $\mu\alpha.\, [\alpha]M = M$ is sound because

$$(\mu\alpha.\, [\alpha]M)^* = \epsilon_{\sigma^*}(\lambda\alpha.\, \alpha(M^*)) = \epsilon_{\sigma^*}(\eta_{\sigma^*}(M^*)) = M^*$$

We prove soundness of $(\mu\alpha^{\sigma_1\to\sigma_2}.\, M)\ N = \mu\beta^{\sigma_2}.\, M[[\beta](-\ N)/[\alpha](-)]$ by using Lemma 2.2. So suppose $f\colon \sigma_2{}^* \multimap R$. Then the function $\lambda^\circ h\colon \sigma_1{}^* \to \sigma_2{}^*.\, f(h(N^*))$ is linear, so by Lemma 2.3

$$\begin{aligned} f((\mu\alpha^{\sigma_1\to\sigma_2}.\, M)\ N)^* &= f(\epsilon_{\sigma_1{}^*\to\sigma_2{}^*}\ (\lambda\alpha\colon (\sigma_1{}^* \to \sigma_2{}^*) \multimap R.\, M^*)(N^*)) \\ &= (\lambda\alpha\colon (\sigma_1{}^* \to \sigma_2{}^*) \multimap R.\, M^*)(\lambda^\circ h\colon \sigma_1{}^* \to \sigma_2{}^*.\, f(h(N^*))) \\ &= M^*[f(-N^*)/\alpha(-)] \\ &= (\lambda\beta\colon \sigma_2{}^* \multimap R.\, (M[[\beta](-\ N)/[\alpha](-)])^*)f \\ &= f(\mu\beta^{\sigma_2}.\, M[[\beta](-\ N)/[\alpha](-)])^*. \end{aligned}$$

Soundness of the last rule $(\mu\alpha^{\forall X.\sigma_1}.\, M)\sigma_2 = \mu\beta^{\sigma_1[\sigma_2/X]}.\, M[[\beta](-\ \sigma_2)/[\alpha](-)]$ is proved the same way. □

Readers familiar with Reus and Streicher's "negated domains" translation of the $\lambda\mu$ calculus [22] may wonder how that translation is related to the one presented here. This question is also relevant for comparison with Hasegawa's work [7] because he studies $\lambda\mu 2$ calculus via a translation which is essentially a polymorphic extension of Reus and Streicher's. In fact, our translation can be seen as a negated domains translation on the value types of the form $\underline{\mathsf{A}} \multimap R$. For each $\lambda\mu 2$ type σ, consider the PE+C value type $\sigma^\dagger =_{\mathrm{def}} \sigma^* \multimap R$. Then $\sigma^\dagger \to R \cong \sigma^*$ and so, up to this isomorphism, any $\lambda\mu 2$ term

$$x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n \vdash M\colon \tau \mid \alpha_1\colon \sigma'_1, \ldots, \alpha_m\colon \sigma'_m$$

is translated to a term

$$x_1\colon \sigma_1{}^\dagger \to R, \ldots, x_n\colon \sigma_n{}^\dagger \to R, \alpha_1\colon \sigma'_1{}^\dagger, \ldots, \alpha_m\colon \sigma'_m{}^\dagger \vdash M^*\colon \tau^\dagger \to R$$

If we add a product to the value types (as can be polymorphically encoded in the presence of parametricity) and an isomorphism $X \cong (X \to R) \multimap R$ for all value types X to PE+C we recover Hasegawa's extension of Reus and Streicher's interpretation since, for example,

$$(\sigma \to \tau)^\dagger \ \cong\ ((\sigma^\dagger \to R) \to (\tau^\dagger \to R)) \multimap R \ \cong\ ((\neg\sigma^\dagger \times \tau^\dagger) \to R) \multimap R \ \cong\ \neg\sigma^\dagger \times \tau^\dagger\ .$$

4 Relationally parametric models of PE+C

The aim of this section is to present a family of relationally parametric models for PE+C. Note that any such model will contain within it a relationally parametric model, in the usual sense, of the standard second-order λ-calculus, since this is a subsystem of the value types of PE. So it is natural to construct the model of PE+C over such a model.

Relationally parametric models of the second-order λ-calculus fall into just two known varieties: term models [8], and PER models. The latter can be viewed explicitly as PER models [1], categorically as fibrations [3], or "synthetically" as full subcategories of the category of "sets" according to the internal logic of associated realizability toposes. Here, as in [14], we take the latter approach, since it allows us to work axiomatically and straightforwardly in intuitionistic set theory, performing simple set-theoretic constructions without getting encumbered by incidental technical properties of the concrete models. Thus, henceforth we work in intuitionistic set theory (IZF). The reader who is unhappy with this, is urged to simply bear with us and to follow the development classically. In theory, following Reynolds [20], such reader will be able to exploit classical logic to derive an inconsistency. However, this awkward fact is unlikely to get in the way of obtaining a general understanding.

We start by recalling the (somewhat involved) structure required for a model of PE in [14]. We assume that we are given a full subcategory $\mathcal{C}$ of the category **Set** of sets that will be used to interpret value types. This is required to satisfy:

(C1) If $A \in \mathcal{C}$ and $A \cong B$ in **Set** then $B \in \mathcal{C}$.

(C2) For any set-indexed family $\{A_i\}_{i\in I}$ of sets in $\mathcal{C}$, the set-theoretic product $\prod_{i\in I} A_i$ is again in $\mathcal{C}$.

(C3) Given $A, B \in \mathcal{C}$ and functions $f, g\colon A \to B$, the equalizer $\{x \in A \mid f(x) = g(x)\}$ is again in $\mathcal{C}$.

(C4) There is a set $\mathbf{C}$ of objects of $\mathcal{C}$ such that, for any $A \in \mathcal{C}$, there exists $B \in \mathbf{C}$ with $B \cong A$.

By items (C2) and (C3), the category $\mathcal{C}$ is complete with limits inherited from **Set**. Since function spaces are powers, for any set A and any $B \in \mathcal{C}$, the function space B^A is in $\mathcal{C}$, i.e. $\mathcal{C}$ is an *exponential ideal* of **Set**. In particular, $\mathcal{C}$ is cartesian closed. The last axiom states that $\mathcal{C}$ is weakly equivalent to a small category. This allows one to show that $\mathcal{C}$ is also cocomplete, by an application of Freyd's adjoint functor theorem, cf. [9].

Computation types are modelled using a category $\mathcal{A}$ together with a functor $U\colon \mathcal{A} \to \mathcal{C}$ satisfying the axioms below. For the moment, the intuition will be that $\mathcal{A}$ is the category of algebras for a monad on $\mathcal{C}$, and U is the forgetful functor. However, soon we shall see a rather different example of model when we consider the case of control. The axioms required for $\mathcal{A}$ and U are:

(A1) U "weakly creates limits" in the following sense. For every diagram Δ in $\mathcal{A}$ and limiting cone $\lim U(\Delta)$ of $U(\Delta)$ in $\mathcal{C}$, there exists a specified limiting cone $\lim \Delta$ of Δ in $\mathcal{A}$ such that $U(\lim \Delta) = \lim U(\Delta)$.

(A2) U reflects isomorphisms (i.e. if Uf is an isomorphism then so is f).

(A3) For objects $\underline{A}, \underline{B}$ of $\mathcal{A}$, the hom-set $\mathcal{A}(\underline{A}, \underline{B})$ is an object of $\mathcal{C}$.

(A4) There exists a set $\mathbf{A}$ of objects of $\mathcal{A}$ such that, for every object $\underline{A}$ of $\mathcal{A}$, there exists $\underline{B} \in \mathbf{A}$ with $\underline{A} \cong \underline{B}$ in $\mathcal{A}$.

Although the above axioms do not force the category $\mathcal{A}$ to be a category of algebras for a monad on $\mathcal{C}$, for convenience and intuition, we shall nonetheless call the objects of $\mathcal{A}$ algebras. Since the axioms imply that U is faithful, we can identify $\mathcal{A}(\underline{A}, \underline{B})$ with a collection of special functions from $U\underline{A}$ to $U\underline{B}$ which we call homomorphisms, and we shall write $f\colon \underline{A} \multimap \underline{B}$ to mean that f is a homomorphism from $U\underline{A}$ to $U\underline{B}$. We write $\underline{A} \cong^{\circ} \underline{B}$ if $\underline{A}$ and $\underline{B}$ are isomorphic as objects of $\mathcal{A}$.

For an object A of $\mathcal{C}$, we write $\mathrm{Sub}_{\mathcal{C}}(A)$ for the set $\{B \subseteq A \mid B \in \mathcal{C}\}$, which gives a canonical representation for the set of $\mathcal{C}$-subobjects of A. Axioms (A1) and (A2) imply that U is an injective map from $\mathcal{A}$-subobjects of $\underline{A} \in \mathcal{A}$ to $\mathcal{C}$-subobjects of $U\underline{A}$. Accordingly, we can define

$$\mathrm{Sub}_{\mathcal{A}}(\underline{A}) =_{\mathrm{def}} \{B \subseteq U\underline{A} \mid B \subseteq U\underline{A} \text{ is the image of an } \mathcal{A}\text{-subobject of } \underline{A}\}$$

as a representation of the set of $\mathcal{A}$-subobjects of $\underline{A}$.

For the construction of the parametric model, we will assume that we are given, as extra data, two collections of "admissible" relations, which we shall use to formulate the principle of relational parametricity. More precisely, for each pair of objects A, B of $\mathcal{C}$, we require a specified set of *admissible $\mathcal{C}$-relations* $\mathcal{R}_{\mathcal{C}}(A, B) \subseteq \mathrm{Sub}_{\mathcal{C}}(A \times B)$, and for each pair of objects $\underline{A}, \underline{B}$ of $\mathcal{A}$, we require a specified set of *admissible $\mathcal{A}$-relations* $\mathcal{R}_{\mathcal{A}}(\underline{A}, \underline{B}) \subseteq \mathrm{Sub}_{\mathcal{A}}(\underline{A} \times \underline{B})$. Moreover, these collections of relations are required to satisfy some closure properties, which we now define.

For $A \in \mathcal{C}$, we write Δ_A for the diagonal (identity) relation in $\mathrm{Sub}_{\mathcal{C}}(A \times A)$. Similarly, for $\underline{A} \in \mathcal{A}$, we write $\Delta_{\underline{A}}$ for the diagonal relation on $U\underline{A}$, which is indeed in $\mathrm{Sub}_{\mathcal{A}}(\underline{A} \times \underline{A})$. For $f\colon A' \to A, g\colon B' \to B$ in $\mathcal{C}$ and $R \in \mathrm{Sub}_{\mathcal{C}}(A \times B)$ we write $(f, g)^{-1}R$ for $\{(x, y) \mid (f(x), g(y)) \in R\}$, which is an element of $\mathrm{Sub}_{\mathcal{C}}(A' \times B')$. Note that if $f\colon \underline{A}' \multimap \underline{A}, g\colon \underline{B}' \multimap \underline{B}$ and $Q \in \mathrm{Sub}_{\mathcal{A}}(\underline{A} \times \underline{B})$ then $(f, g)^{-1}Q$ is an element of $\mathrm{Sub}_{\mathcal{A}}(\underline{A}' \times \underline{B}')$.

We impose the following requirements on admissible relations.

(R1) For each object A of $\mathcal{C}$ the diagonal relation Δ_A is in $\mathcal{R}_{\mathcal{C}}(A, A)$ and likewise for each object $\underline{A}$ of $\mathcal{A}$ the diagonal $\Delta_{\underline{A}}$ is in $\mathcal{R}_{\mathcal{A}}(\underline{A}, \underline{A})$.

(R2) Admissible relations are closed under inverse image, i.e., if $R \in \mathcal{R}_{\mathcal{C}}(A, B)$ and $f\colon A' \to A$, $g\colon B' \to B$, then $(f, g)^{-1}R \in \mathcal{R}_{\mathcal{C}}(A', B')$ and if $Q \in \mathcal{R}_{\mathcal{A}}(\underline{A}, \underline{B})$ and $f\colon \underline{A}' \multimap \underline{A}$, $g\colon \underline{B}' \multimap \underline{B}$, then $(f, g)^{-1}Q \in \mathcal{R}_{\mathcal{A}}(\underline{A}', \underline{B}')$

(R3) For any set of admissible $\mathcal{C}$- (respectively $\mathcal{A}$-)relations on the same pair of objects, the intersection is an admissible $\mathcal{C}$- (respectively $\mathcal{A}$-)relation.

(R4) $\mathcal{R}_{\mathcal{A}}(\underline{A}, \underline{B}) \subseteq \mathcal{R}_{\mathcal{C}}(U\underline{A}, U\underline{B})$.

Notice that axioms $(R1)$ and $(R2)$ imply that graphs of functions are admissible, specifically if $f\colon A \to B$ then $\langle f \rangle =_{\mathrm{def}} \{(x, y) \mid f(x) = y\} \in \mathcal{R}_{\mathcal{C}}(A, B)$ and if $g\colon \underline{A} \multimap \underline{B}$ then $\langle g \rangle \in \mathcal{R}_{\mathcal{A}}(\underline{A}, \underline{B})$.

There are many ways of constructing models for axioms (C), (A) and (R). A motivating construction is to take $\mathcal{C}$ to be any full reflective subcategory of a real-

izability topos that is weakly small (see [9,11] for how to produce such categories). Then for any (internal) monad on $\mathcal{C}$, take $\mathcal{A}$ to be the category of (Eilenberg-Moore) algebras for the monad. Finally, take $\mathcal{R}_{\mathcal{C}}(A,B)$ and $\mathcal{R}_{\mathcal{A}}(\underline{A},\underline{B})$ to simply be $\mathrm{Sub}_{\mathcal{C}}(A\times B)$ and $\mathrm{Sub}_{\mathcal{A}}(\underline{A}\times\underline{B})$ respectively. A more sophisticated example, exploiting the flexibility in specifying the admissible relations, appears below.

We briefly consider the interpretation of PE in the above structure, giving an overview sufficient for understanding this paper. For full details see [14].

Value types are interpreted in the category $\mathcal{C}$ and computation types are interpreted as objects in the category $\mathcal{A}$. We shall write $\mathcal{C}[\![-]\!]$ and $\mathcal{A}[\![-]\!]$ for these interpretations respectively. Since any computation type $\underline{\mathsf{A}}$ is also a value type, it is given two interpretations which satisfy $U(\mathcal{A}[\![\underline{\mathsf{A}}]\!])=\mathcal{C}[\![\underline{\mathsf{A}}]\!]$.

The type constructor $\to$ is interpreted as set-theoretic function space and $\multimap$ is interpreted as the set of homomorphisms between the algebras assigned to the computation types. The computation type $\mathsf{A}\to\underline{\mathsf{B}}$ is interpreted, using weak limit creation, as an algebra $\underline{A}$ for which $U\underline{A}$ is the set of functions from $\mathcal{C}[\![\mathsf{A}]\!]$ to $\mathcal{C}[\![\underline{\mathsf{B}}]\!]$. In order to give a relationally parametric interpretation of polymorphic types, the type constructors are also given relational interpretations. For example, if $R\in\mathcal{R}_{\mathcal{C}}(A,B)$ and $R'\in\mathcal{R}_{\mathcal{C}}(A',B')$ then $R\to R'$ is the relation

$$\{(f,g)\in(A\to A')\times(B\to B')\mid\forall x\colon A,y\colon B.\,(x,y)\in R\supset(f(x),g(y))\in R'\},$$

and if $Q\in\mathcal{R}_{\mathcal{A}}(\underline{A},\underline{B})$ and $Q'\in\mathcal{R}_{\mathcal{A}}(\underline{A}',\underline{B}')$ then $Q\multimap Q'$ is the relation

$$\{(f,g)\in(\underline{A}\multimap\underline{A}')\times(\underline{B}\multimap\underline{B}')\mid\forall x\colon U\underline{A},y\colon U\underline{B}.\,(x,y)\in Q\supset(f(x),g(y))\in Q'\}.$$

Polymorphic types are interpreted by taking products over $\mathbf{C}$ and $\mathbf{A}$ respectively, and restricting to parametric elements of the product. This restriction to parametric elements is with respect to parametricity formulated with respect to relations in $\mathcal{R}_{\mathcal{C}}$ when quantifying over value types and with respect to relations in $\mathcal{R}_{\mathcal{A}}$ when quantifying over computation types. As an illustrative example, which we shall need later, for a closed computation type $\underline{\mathsf{B}}$, the type $\forall\underline{X}.\,((\underline{X}\multimap\underline{\mathsf{B}})\to\underline{\mathsf{B}})\multimap\underline{X}$ is interpreted as the object of $\mathcal{A}$ given by

$$\begin{aligned}\{(f_{\underline{A}})_{\underline{A}\in\mathbf{A}}\in\textstyle\prod_{\underline{A}\in\mathbf{A}}((\underline{A}\multimap\mathcal{A}[\![\underline{\mathsf{B}}]\!])\to\mathcal{A}[\![\underline{\mathsf{B}}]\!])\multimap\underline{A}\mid\forall\underline{A},\underline{B}\in\mathbf{A}.\,\forall Q\in\mathcal{R}_{\mathcal{A}}(\underline{A},\underline{B}).\\ (f_{\underline{A}},f_{\underline{B}})\in((Q\multimap\Delta_{\mathcal{A}[\![\underline{\mathsf{B}}]\!]})\to\Delta_{\mathcal{A}[\![\underline{\mathsf{B}}]\!]})\multimap Q\}.\end{aligned}\tag{1}$$

Finally, we briefly recall the interpretation of terms. For simplicity we restrict to the case in which types are closed. A well-typed term $\Gamma\mid\Delta\vdash t\colon\mathsf{B}$ is interpreted as an element $[\![t]\!]_\gamma$ of $\mathcal{C}[\![\mathsf{B}]\!]$ relative to an environment γ mapping each variables $x\colon\mathsf{A}$ in $\Gamma\mid\Delta$ to an element $\gamma(x)\in\mathcal{C}[\![\mathsf{A}]\!]$. In the case of Δ being non-empty, the mapping $d\in\mathcal{C}[\![\underline{\mathsf{A}}]\!]\mapsto[\![t]\!]_{\gamma[d/x]}$ is a homomorphism from $\mathcal{A}[\![\underline{\mathsf{A}}]\!]$ to $\mathcal{A}[\![\underline{\mathsf{B}}]\!]$ for any environment γ. For the full interpretation of terms involving open types, see [14].

Having now presented the semantic structure needed to model PE, we at last turn to the issue of, in addition, modelling our constructs for control. Since there is no avoiding having a parametric model of second-order λ-calculus, we begin with an arbitrary category $\mathcal{C}$ satisfying the (C) axioms above. Let R be any object of $\mathcal{C}$,

chosen as a result type. In order to satisfy our axioms, one would like to implement the equation:

$$\mathcal{A} \xrightarrow{U} \mathcal{C} \quad =_{\mathrm{def}} \quad \mathcal{C}^{\mathrm{op}} \xrightarrow{R^{(-)}} \mathcal{C} \ ,$$

which fits in with an established tradition in continuations models, cf. [25,24,13]. Indeed, in such a structure, one would interpret the computation type constant R by $\mathcal{A}[\![R]\!] =_{\mathrm{def}} 1$ and $\mathcal{C}[\![R]\!] =_{\mathrm{def}} R^1 \cong R$. Then, for any object $\underline{A}$ of $\mathcal{A}$, and hence of $\mathcal{C}$, one would have $\mathcal{A}[\![(\underline{A} \multimap R) \to R]\!] \cong^{\circ} \underline{A}$,because $\mathcal{C}[\![\underline{A} \multimap R]\!] \cong \mathcal{A}(\underline{A}, 1) = \mathcal{C}(1, \underline{A})$, hence $\mathcal{C}[\![(\underline{A} \multimap R) \to R]\!] \cong R^{\underline{A}} = U(\underline{A})$, where the isomophism is indeed a homomorphism.

However, although the category $\mathcal{C}^{\mathrm{op}}$ does satisfy axioms (A3) and (A4) above, the functor $R^{(-)} \colon \mathcal{C}^{\mathrm{op}} \to \mathcal{C}$ does not necessarily satisfy axioms (A1) and (A2). Accordingly, we modify the construction so that these axioms are satisfied.

First we address the reflection of isomorphisms. Although $R^{(-)} \colon \mathcal{C}^{\mathrm{op}} \to \mathcal{C}$ need not reflect isomorphisms on all of $\mathcal{C}$, there exists a largest full reflective subcategory of $\mathcal{C}$ on which it does.

Definition 4.1 [10,23] A set C in $\mathcal{C}$ is called *R-replete* if for all maps $f \colon A \to B$ in $\mathcal{C}$, if the map $R^f \colon R^B \to R^A$ is an isomorphism then so is $C^f \colon C^B \to C^A$.

We denote by $\mathcal{C}_{\mathrm{rep}}$ the full subcategory of $\mathcal{C}$ on replete objects. It is standard, cf. [10,23], that the replete objects are closed under limits in **Set** and so $\mathcal{C}_{\mathrm{rep}}$ satisfies (C1)–(C3). It also satisfies (C4) as we can take $\mathbf{C}_{\mathrm{rep}} =_{\mathrm{def}} \{C \in \mathbf{C} \mid C \text{ replete}\}$. Trivially the object R is itself replete, and hence any power R^A is also replete.

As a second attempt at constructing a model we replace (4) with:

$$\mathcal{A} \xrightarrow{U} \mathcal{C}_{\mathrm{rep}} \quad =_{\mathrm{def}} \quad {\mathcal{C}_{\mathrm{rep}}}^{\mathrm{op}} \xrightarrow{R^{(-)}} \mathcal{C}_{\mathrm{rep}} \ ,$$

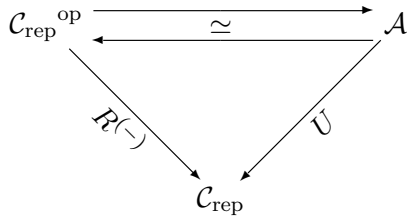
However, there remains a technical issue concerning satisfaction of (A1). As remarked earlier, the axioms (C1)–(C4) imply cocompleteness, and so ${\mathcal{C}_{\mathrm{rep}}}^{\mathrm{op}}$ is complete. Moreover, the functor $R^{(-)}$ preserves limits since it has a left adjoint (also given by $R^{(-)}$). Still $R^{(-)} \colon {\mathcal{C}_{\mathrm{rep}}}^{\mathrm{op}} \to \mathcal{C}_{\mathrm{rep}}$ does not weakly create limits in the sense of (A1), since a given limiting cone in $\mathcal{C}_{\mathrm{rep}}$ of a diagram of the form R^{Δ} need not be in the image of the functor $R^{(-)}$ up to equality. We address this by considering instead of ${\mathcal{C}_{\mathrm{rep}}}^{\mathrm{op}}$ the equivalent category $\mathcal{A}$ (and this is our final redefinition of $\mathcal{A}$):

Objects: Triples (A, i, P), where A, P are objects in $\mathcal{C}_{\mathrm{rep}}$ and $i \colon P \to R^A$ is an isomorphism.

Morphisms: A morphism from (A, i, P) to (B, j, Q) is a map $f \colon B \to A$.

We define the functor $U \colon \mathcal{A} \to \mathcal{C}$ to map the morphism $f \colon (A, i, P) \to (B, j, Q)$ to $j^{-1} \circ R^f \circ i \colon P \to Q$. Clearly there exists an equivalence of categories between ${\mathcal{C}_{\mathrm{rep}}}^{\mathrm{op}}$

and $\mathcal{A}$ making the diagram



commute up to natural isomorphism, but $U\colon \mathcal{A} \to \mathcal{C}_{\mathrm{rep}}$ does have the property of weakly creating limits in the sense of (A1).

Proposition 4.2 *The functor $U\colon \mathcal{A} \to \mathcal{C}_{\mathrm{rep}}$ satisfies (A1)–(A4).*

Before discussing admissible relations in this model we write out the interpretation of the non-polymorphic types of PE+C. If $\mathcal{A}[\![\underline{\mathsf{A}}]\!] = (A, i, \mathcal{C}[\![\underline{\mathsf{A}}]\!])$ and $\mathcal{A}[\![\underline{\mathsf{B}}]\!] = (B, j, \mathcal{C}[\![\underline{\mathsf{B}}]\!])$ then

$$\begin{aligned}
\mathcal{A}[\![R]\!] &= (1, R \cong R^1, R)\\
\mathcal{C}[\![\mathsf{A} \to \mathsf{B}]\!] &= \mathcal{C}[\![\mathsf{B}]\!]^{\mathcal{C}[\![\mathsf{A}]\!]}\\
\mathcal{C}[\![\underline{\mathsf{A}} \multimap \underline{\mathsf{B}}]\!] &= A^B\\
\mathcal{A}[\![\mathsf{A} \to \underline{\mathsf{B}}]\!] &= (B \times \mathcal{C}[\![\mathsf{A}]\!], h, \mathcal{C}[\![\underline{\mathsf{B}}]\!]^{\mathcal{C}[\![\mathsf{A}]\!]})
\end{aligned}$$

where h is $j^{\mathcal{C}[\![\mathsf{A}]\!]}$ composed with the isomorphism $(R^B)^{\mathcal{C}[\![\mathsf{A}]\!]} \cong R^{B \times \mathcal{C}[\![\mathsf{A}]\!]}$.

Finally, we consider how to define admissible relations. The type theory PE+C introduces the isomorphism $(\underline{X} \multimap R) \to R \cong^{\circ} \underline{X}$ via a *polymorphic* constant of type $\forall \underline{X}.\, ((\underline{X} \multimap R) \to R) \multimap \underline{X}$ inverse to η. By (1), for this inverse to be parametric, we must have, for any $Q \in \mathcal{R}_{\mathcal{A}}(\underline{A} \times \underline{B})$,

$$(\eta_{\underline{A}}, \eta_{\underline{B}})^{-1}((Q \multimap \Delta_R) \to \Delta_R) = Q. \tag{2}$$

This forces us to consider, as our notion of admissible relation, the collection of relations Q satisfying (2). Fortunately, $(\eta_{\underline{A}}, \eta_{\underline{B}})^{-1}((Q \multimap \Delta_R) \to \Delta_R)$ has a familiar looking alternative description as $Q^{\top\top}$ defined as

$$\begin{aligned}
Q^{\top} &= \{(f, g)\colon (\underline{A} \multimap R) \times (\underline{B} \multimap R) \mid \forall x\colon U\underline{A}, y\colon U\underline{B}.\, (x, y) \in Q \supset f(x) = g(y)\}\\
Q^{\top\top} &= \{(x, y) \in U\underline{A} \times U\underline{B} \mid \forall f\colon \underline{A} \multimap R, g\colon \underline{B} \multimap R.\, (f, g) \in Q^{\top} \supset f(x) = g(y)\}
\end{aligned}$$

The $(-)^{\top\top}$ construction defines a closure operator in the sense that it is increasing ($Q \subseteq Q^{\top\top}$), idempotent and monotone with respect to the inclusion ordering, and we shall call a relation $(-)^{\top\top}$-closed if $Q^{\top\top} = Q$. The $(-)^{\top\top}$-closure is a familiar construction in the theory of parametricity, studied extensively by Pitts in an operational setting [17,2]. The technique also appears to be a general and powerful one from a denotational perspective, cf. [12].

Finally, we define:

$$\mathcal{R}_{\mathcal{C}}(A,B) = \mathrm{Sub}_{\mathcal{C}}(A \times B)$$
$$\mathcal{R}_{\mathcal{A}}(\underline{A},\underline{B}) = \{Q \in \mathrm{Sub}_{\mathcal{A}}(\underline{A} \times \underline{B}) \mid Q = Q^{\top\top}\}$$

Proposition 4.3 *Axioms (R1)–(R4) for admissible relations hold.*

Proof. The interesting point is that, for any $\underline{A} \in \mathcal{A}$, the identity relation $\Delta_{\underline{A}}$ is $(-)^{\top\top}$ closed. This is easily seen to be equivalent to the following linear separation property: for all $x, y \in U\underline{A}$, if $f(x) = f(y)$ for all $f\colon \underline{A} \multimap R$, then $x = y$; and this, in turn, is equivalent to the canonical function $\eta_{\underline{A}}\colon \underline{A} \multimap (\underline{A} \multimap R) \to R$ being injective. However, we showed above that this function is an isomorphism. □

Theorem 4.4 *The structure defined above is a model of* PE+C.

Proof. Most of this theorem follows from Propositions 4.2 and 4.3. It remains to prove that this model of PE also models the polymorphic inverse to η. Since each $\eta_{\underline{A}}$ has an inverse $\epsilon_{\underline{A}}$ we just need to show that this collection constitutes an element in the parametric model, i.e., that for any $(-)^{\top\top}$ closed relation Q between algebras $\underline{A}, \underline{B}$, the pair $(\epsilon_{\underline{A}}, \epsilon_{\underline{B}})$ maps elements related in $(Q \multimap \Delta_R) \to \Delta_R$ to elements related in Q. But this happens iff $(\eta_{\underline{A}}, \eta_{\underline{B}})^{-1}((Q \multimap \Delta_R) \to \Delta_R) = Q$ which holds for $(-)^{\top\top}$ closed relations Q since $Q^{\top\top} = (\eta_{\underline{A}}, \eta_{\underline{B}})^{-1}((Q \multimap \Delta_R) \to \Delta_R)$. □

5 Polymorphic type encodings in $\lambda\mu 2$

By composing the interpretation from $\lambda\mu 2$ into PE+C with the interpretation of PE+C into any model as defined in the previous section, we get a parametric model of $\lambda\mu 2$. In this section we study polymorphic $\lambda\mu 2$ type encodings in such models, and compare with Hasegawa's parametricity results for $\lambda\mu 2$ [7].

The composite translation interprets types σ of $\lambda\mu 2$ as objects $\mathcal{A}[\![\sigma^*]\!]$ in $\mathcal{A}$, but terms $t\colon \sigma \to \tau$ of $\lambda\mu 2$ are interpreted not as homomorphisms but as general maps from $U(\mathcal{A}[\![\sigma^*]\!])$ to $U(\mathcal{A}[\![\tau^*]\!])$ (recalling that $U(\mathcal{A}[\![\underline{\mathsf{A}}]\!]) = \mathcal{C}[\![\underline{\mathsf{A}}]\!]$). For the following discussion it is therefore convenient to introduce the category $\mathcal{A}_!$, with the same objects as $\mathcal{A}$, i.e., triples (A, i, P) consisting of replete objects A, P and an isomorphism $i\colon P \to R^A$. A morphism from (A, i, P) to (B, j, Q) is simply a map from P to Q. Note that the category $\mathcal{A}_!$ is equivalent to the full subcategory of **Set** on the objects of the form R^A for A in $\mathcal{C}_{\mathrm{rep}}$. This is called the category of "negated domains" by Reus and Streicher [22], and indeed our interpretation of $\lambda\mu 2$ in $\mathcal{A}_!$ can be seen as a polymorphic extension of their interpretation of (simply-typed) $\lambda\mu$. Alternatively, one can view our interpretation as a polymorphic extension of Selinger's [21], since $\mathcal{A}_!$ is a *control category* in the sense of *op. cit.*

The category $\mathcal{A}$ embeds into $\mathcal{A}_!$ by mapping a morphism from (A, i, P) to (B, j, Q) given by $f\colon B \to A$ to $j^{-1} \circ R^f \circ i$. We shall call a term $t\colon \sigma \to \tau$ of $\lambda\mu 2$ *linear* if it is interpreted in the model as a homomorphism.

Our goal in this section is to uncover the universal properties of polymorphic type encodings in $\lambda\mu 2$ that are induced by our relationally parametric model. All

the examples we consider are taken from Hasegawa's paper [7]. We obtain the same universal properties with respect to the category of linear maps that he obtains for his "focal" maps. The main point we wish to emphasise is that our results fall out as instances of general definability results valid for any model of PE, and hence as instances of principles that are valid for arbitrary computational effects [14].

Example 5.1 The $\lambda\mu 2$ type $\forall X.\, X$ is interpreted as an initial object in $\mathcal{A}$. Indeed, for any type σ, the term $\lambda x\colon \forall X.\, X.\, x\,\sigma$ defines the unique linear map from $\forall X.\, X$ to σ. This is an immediate consequence of the following general fact, cf. [14]. In any parametric model of PE, the computation type $\forall \underline{X}.\, \underline{X}$ is interpreted as the initial object in $\mathcal{A}$, with the unique linear map given in the evident way.

Note that the $\lambda\mu 2$ type $\bot$ is also interpreted as an initial object in $\mathcal{A}$ (the object $(1, R \cong R^1, R)$ is obviously initial). So relational parametricity yields the polymorphic definability of $\bot$ in $\lambda\mu 2$. This fact can be used to justify omitting $\bot$ as a primitive type constant in $\lambda\mu 2$, and using the polymorphic type $\forall X.\, X$ in its place, as, in fact, is done in [7].

An analogous observation applies to the formulation of PE+C. In any parametric model of PE+C, the type constant R is interpreted as the initial object of $\mathcal{A}$. Thus the constant R could have simply been omitted from PE+C, and the isomorphisms formulated using the computation type $\forall \underline{X}.\, \underline{X}$ in its place.

Example 5.2 The $\lambda\mu 2$ type $\forall X.\, (\sigma \to X) \to X$, where X is not free in σ, is given an interpretation that is linearly isomorphic to the interpretation of $\neg\neg\sigma$. In $\lambda\mu 2$, the canonical terms of type $\sigma \to \forall X.\, (\sigma \to X) \to X$ and $(\forall X.\, (\sigma \to X) \to X) \to \sigma$ are not isomorphisms. Rather they respectively correpond to the unit for the double negation monad and double negation elimination in the control category $\mathcal{A}_!$.

The above properties follow from the general fact [14] that, in any parametric model of PE, the type $!\mathsf{A} = \forall \underline{X}.\, (\mathsf{A} \to \underline{X}) \to \underline{X}$ is interpreted as the free $\mathcal{A}$-object over the set $\mathcal{C}[\![\mathsf{A}]\!]$. In any model of PE+C, this specialises to $\mathcal{C}[\![!\mathsf{A}]\!] \cong^\circ \neg\neg\mathcal{C}[\![\mathsf{A}]\!]$, cf. the motivating discussion for the control isomorphisms in Section 2.

Example 5.3 Suppose σ is a type expression of $\lambda\mu 2$ with type variable X occuring only positively. As is standard, this type induces an endofunctor on $\mathcal{A}_!$. One can check, by induction on the structure of σ, that this functor cuts down to one acting on the category of linear maps $\mathcal{A}$.

In the pure second-order λ-calculus, the type $\forall X.\, (\sigma \to X) \to X$ is an initial algebra for the functor induced by σ. However, in our model of $\lambda\mu 2$, the type $\forall X.\, (\sigma \to X) \to X$ is instead an initial algebra for the functor induced by the type $\neg\neg\sigma$ on the linear category $\mathcal{A}$. Moreover, all the necessary structure is definable in $\lambda\mu 2$, i.e., the structure map of the initial algebra can be defined as a linear map in $\lambda\mu 2$, and likewise the term giving the unique algebra map out of a given algebra. Detailed definitions appear in [7].

The above facts are a conseqence of the following general property of PE models, cf. [14]. For any computation type $\underline{\mathsf{A}}$ with free computation type variable $\underline{X}$ occurring only positively, the induced functor on $\mathcal{A}$, has its initial algebra $\mu^\circ \underline{X}.\, \underline{\mathsf{A}}$ given by the computation type $\forall \underline{X}.\, (\underline{\mathsf{A}} \multimap \underline{X}) \to \underline{X}$. Then in PE+C we have $(\forall X.\, (\sigma \to X) \to X)^* = \forall \underline{X}.\, (\sigma^* \to \underline{X}) \to \underline{X}$, which is isomorphic to $\mu^\circ \underline{X}.\, \neg\neg\sigma^* =$

$$\begin{aligned}
\mathcal{C}[\![(\forall X.\, X \to X)^*]\!] &\cong^\circ R^R \\
\mathcal{C}[\![(\forall X.\, (\sigma \to X) \to (\tau \to X) \to X)^*]\!] &\cong^\circ R^{R^{\mathcal{C}[\![\sigma^*]\!]} \times R^{\mathcal{C}[\![\tau^*]\!]}} && (X \notin \mathrm{FTV}(\sigma,\tau)) \\
\mathcal{C}[\![(\forall X.\, (\sigma \to \tau \to X) \to X)^*]\!] &\cong^\circ R^{R^{\mathcal{C}[\![\sigma^*]\!]}} \times R^{R^{\mathcal{C}[\![\tau^*]\!]}} && (X \notin \mathrm{FTV}(\sigma,\tau)) \\
\mathcal{C}[\![(\forall Y.\, (\forall X.\, (\sigma \to Y)) \to Y)^*]\!] &\cong^\circ \mathcal{C}[\![\exists^\circ \underline{X}.\, \neg\neg\sigma^*]\!] && (Y \notin \mathrm{FTV}(\sigma))
\end{aligned}$$

Fig. 6. Interpretation of $\lambda\mu 2$ types in the model

$\forall \underline{X}.\, (\neg\neg\sigma^* \multimap \underline{X}) \to \underline{X}$.

A few more examples of interpretations of $\lambda\mu 2$ types are presented in Figure 6, where the notation $\exists^\circ \underline{X}.\, \underline{\mathsf{A}}$ represents an existential computation type, see [14]. The stated isomorphisms again follow from general properties of PE models.

While the results we have presented above all follow those of Hasegawa [7], there is a difference in the formulation. We have obtained universal properties relative to our semantic notion of linear map, whereas Hasegawa obtained them with respect to a syntactically defined notion of "focal map". Here the semantic/syntactic distinction is not crucial. We could equally well have used our translation from $\lambda\mu 2$ to PE+C to define a syntactic notion of linear map in $\lambda\mu 2$ (and, in effect, this is anyway how our proofs that various terms were linear proceed). Conversely, one can immediately apply the definition of focal map in our semantic setting to obtain a subcategory of focal maps within $\mathcal{A}_!$. Doing this, one obtains that every linear map is focal, but the converse holds if and only if the continuation monad $R^{R^{(-)}}$ on $\mathcal{C}$ satisfies Moggi's equalising requirement [15]. The equalising requirement seems a difficult condition to enforce for continuations monads in semantic categories modelling parametric polymorphism. For example, in [5], it is shown that the equalising requirement can fail in the category of R-replete objects. Thus, we feel that our stricter notion of linear map is the more natural one in our semantic setting.

Acknowledgements

We thank Masahito Hasegawa and Paul Levy for helpful discussions.

References

[1] E.S. Bainbridge, P.J. Freyd, A. Scedrov, and P.J. Scott. Functorial polymorphism. *Theoretical Computer Science*, 70:35–64, 1990.

[2] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In *Fourth International Workshop on Higher Order Operational Techniques in Semantics, Montréal*, volume 41 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 2000.

[3] L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*, 15(4):709–772, 2005.

[4] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Linear Abadi & Plotkin logic. *Logical Methods in Computer Science*, 2, 2006.

[5] G. Gruenhage and T. Streicher. Quotients of countably based spaces are not closed under sobrification. *Math. Struct. in Comp. Sci.*, 16:223–229, 2006.

[6] M. Hasegawa. Relational parametricity and control. In *Proceedings Twentieth Annual LiCS Symposium*, pages 72–81, 2005.

[7] M. Hasegawa. Relational parametricity and control. *Logical Methods in Computer Science*, 2, 2006.

[8] R. Hasegawa. Parametricity of extensionally collapsed term models of polymorphism and their categorical properties. In Takayasu Ito and Albert R. Meyer, editors, *Proceedings of Theoretical Aspects of Computer Software.(TACS '91)*, volume 526 of *LNCS*, pages 495–512. Springer, September 1991.

[9] J.M.E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40(2):135–165, November 1988.

[10] J.M.E. Hyland. First steps in synthetic domain theory. In A. Carboni, M.C. Pedicchio, and G. Rosolini, editors, *Proceedings of the 1990 Como Category Theory Conference*, volume 1488 of *Lecture Notes in Mathematics*, pages 131–156. Springer, Berlin, 1991.

[11] J.M.E. Hyland, E.P. Robinson, and G. Rosolini. The discrete objects in the effective topos. *Proc. London Math. Soc.*, 3(60):1–36, 1990.

[12] S. Katsumata. A Semantic Formulation of $\top\top$-lifting and Logical Predicates for Computational Metalanguage. In *Proc. CSL 2005.* LNCS 3634, pp. 87–102, 2005.

[13] P. Levy. *Call By Push Value, a Functional/ Imperative Synthesis.* Kluwer, December 2003.

[14] R.E. Møgelberg and A.K. Simpson. Relational parametricity for computational effects. Submited manuscript, 2007.

[15] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[16] M. Parigot. Proofs of strong normalisation for second order classical natural deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, 1997.

[17] A.M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in computer Science*, 10:321–359, 2000.

[18] G.D. Plotkin. Type theory and recursion (extended abstract). In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, page 374, Montreal, Canada, 19–23 June 1993. IEEE Computer Society Press.

[19] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983.

[20] J.C. Reynolds. Polymorphism is not set-theoretic. In G. Kahn, D. B. MacQueen, and G. D. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 145–156. Springer-Verlag, 1984.

[21] P. Selinger. Control categories and duality: On the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.

[22] T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, 1998.

[23] P. Taylor. The fixed point property in synthetic domain theory. In *6th Annual Symposium on Logic in Computer Science*, pages 152–160, Washington, 1991. IEEE Computer Society Press.

[24] P. Taylor. Sober spaces and continuations. *Theory and Applications of Categories*, 10(12):248–300, 2002.

[25] H. Thielecke. *Categorical Structure of Continuation Passing Style.* Phd thesis, Laboratory for Foundations of Computer Science, Univ. of Edinburgh, 1997.